REMARKS

Claims 1, 6, 7 and 27-38 have been amended. No claims have been added or cancelled. Therefore, claims 1-38 remain pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Section 112, Second Paragraph, Rejection:

The Examiner rejected claims 1-38 under 35 U.S.C. § 112, second paragraph, as being indefinite. Specifically, the Examiner asserts that claims 1, 15 and 27 are unclear at to which requests are being sent to the primary scheduler. Claim 1 has been amended according to the Examiner's suggestion in the Response to Arguments section. However, Applicants assert that this amendment is redundant with respect to the original claim language and is made solely to appease the Examiner, not to change the scope of the claim or for any reason of patentability. Both claims 15 and 27 recite, in part, "receiving a plurality of management requests", "scheduling the plurality of management requests ... after receiving the management requests", "sending the management requests" and "executing the management requests". Thus, both claim 15 and claim 27 are quite clear regarding which request are being sent.

The Examiner also argues that claims 6 and 7 are unclear as to which requests are being sent. Claims 6 and 7 have been amended to appease the Examiner. The amendments to claims 6 and 7 are redundant with respect to the original claim language and do not change the scope of the claim and are not made for any reason of patentability.

Applicants request removal of the Section 112 rejection.

Section 103(a) Rejection:

The Examiner rejected claims 1-38 under 35 U.S.C. § 103(a) as being unpatentable over Kimmel et al. (U.S. Patent 6,105,053) (hereinafter "Kimmel") in view

of Maresco (U.S. Patent 6,418,458). Applicants respectfully traverse the rejection of claims 1-38 for at least the following reasons.

Regarding claim 1, contrary to the Examiner's assertion, Kimmel in view of Maresco fails to teach or suggest a primary scheduler that is executable to schedule requests for networked data resources. The Examiner cites column 24, lines 14-26 and column 6, lines 43-65 of Kimmel. Kimmel teaches an operating system for non-uniform memory access (NUMA) multiprocessor systems and utilizes a software abstraction of the system hardware to maintain balanced processor and memory loads (Kimmel, Abstract). The Examiner's first cited passage (column 24, lines 14-26) describes how Kimmel's operating system may function in systems having various numbers of execution queue levels and that have differently sized thread groups. The second passage cited by the Examiner describes the thread group structure maintaining cumulative timeslice and job processor (JP) accounting for all threads in a thread group. This cited passage also describes how a JP dispatcher selects a thread group to execute. Kimmel teaches that a dispatcher selects an individual thread from a thread group based on the local priority and scheduling policy. Selection occurs at two levels. Global scheduling policies are used to select a thread group, while local scheduling policies are used to select an individual thread from the selected thread group.

Neither of the Examiner's cited passages teaches or suggests a primary scheduler executable to schedule *requests for networked data resources*. Kimmel's JP dispatcher selects a thread to execute on a processor of the NUMA system. Kimmel fails to mention any networked data resources. All resources taught by Kimmel are local to a single NUMA system and are not networked data resources. Additionally, Kimmel makes no mention of *requests* for networked data resources. Kimmel states that the "dispatcher is a kernel subsystem that is a mechanism responsible for scheduling and executing processes on an associated JP [job processor] in accordance with certain global and local scheduling policies" (Kimmel, column 5, lines 43-46). Kimmel is concerned with balanced processor and memory loads in a NUMA system. Kimmel does not pertain to *scheduling requests for networked data resources*.

Maresco pertains to the creation of threads to prioritize the execution of tasks in an object oriented system (Maresco, column 1, lines 42-57; column 2, lines 25-38). **Neither Kimmel nor Maresco has anything to do with scheduling requests for networked data resources.** All resources taught by Kimmel are local to a single NUMA system and are not networked data resources. And the threads in Maresco are clearly not requests for networked data resources. Thus, Kimmel and Maresco clearly do not teach or suggest anything about scheduling requests for networked data resources.

In the Response to Arguments, the Examiner reasserts that Kimmel "teaches request [*sic*] for networked data requests", citing the abstract, column 1, lines 30-55, column 3, lines 1-15, column 4, lines 39-65, column 24, lines 14-26, and column 6, lines 43-65 of Kimmel. The Examiner fails to respond to the previous arguments concerning the cited portions of columns 24 and 6. The cited portion of column 1 is from Kimmel's background section describing a separate system, which "discloses a method for affining groups of related threads from the same process to a group of JPs to improve secondary cache affinity while improving efficiency of operations among threads in the same group and reducing overhead for operations between groups." The cited passage further discloses the "need for an operating system having a global scheduling mechanism that may be implemented in a scalable multiprocessor system having a NUMA architecture." Contrary to the Examiner's assertion, column 1, lines 30-55 of Kimmel does not mention anything about requests for networked data resources. As stated above, Kimmel fails to mention any networked data resources. All resources taught by Kimmel are local to a single NUMA system and are not networked data resources.

Additionally, Kimmel makes no mention of *requests* for networked data resources. Kimmel is concerned with balanced processor and memory loads in a NUMA system. Kimmel is not concerned with *scheduling requests for networked data resources*. In the portion of column 3 cited by the Examiner, Kimmel describes that "each run queue that is associated with one of the remaining nodes identif[ies] the active process affined to groups of the processors that share the resource associated with the

node." As stated shown above, this relates to a single multi-processor system and nowhere describes *scheduling requests for networked data resources*. The portion of column 4 cited by the Examiner refers to JP caches, and, as shown above, also does not have anything to do with *scheduling requests for networked data resources*. Likewise, Kimmel's abstract is completely devoid of any mention of scheduling requests for networked data resources.

Similarly, the Examiner's cited portions of columns 24 and 6 also fail to mention anything at all about requests for networked data requests." Instead, column 6, lines 43-65 describe Kimmel's method for the JP's dispatcher to select a thread group to execute and column 24, lines 14-26 is merely a statement by Kimmel that his invention may be implemented in forms other than those specifically described by Kimmel. As with the Examiner's other cited passages, neither the cited portion of column 6 or of column 24 mention anything about requests for networked data resources as the Examiner contends.

In further regard to claim 1, contrary to the Examiner's assertion, Kimmel in view of Maresco also fails to teach or suggest a secondary scheduler, wherein the secondary scheduler is executable to receive a plurality of requests from a multi-threaded application in a thread-safe manner and send the requests to the primary scheduler in a thread-safe manner. The Examiner cites several passages of Kimmel referring to Kimmel's medium term scheduler. However, Kimmel's medium term scheduler does not receive any requests from a multi-threaded application. Instead, Kimmel's medium term scheduler "monitors the progress of active processes in the system and sets a flag for those processes that are not progressing" (Kimmel, column 2, lines 40-45). Nowhere does Kimmel describe his medium term scheduler receiving requests from a multi-threaded application. Furthermore, the medium term scheduler in Kimmel's system does not send requests received from a multi-threaded application to Kimmel's dispatcher, which the Examiner equates to the primary scheduler of claim 1. The Examiner has not pointed out any aspect or feature of Kimmel's system that can be interpreted as a secondary scheduler executable to *receive requests from a multi-threaded application* and to *send the requests to a primary scheduler*.

Instead, Kimmel's medium term scheduler monitors thread groups to identify languishing thread groups and candidates for load balancing by monitoring the loads of the respective scheduling locals (i.e. certain nodes in Kimmel's hierarchical representation of a system's hardware) to identify any load imbalances and by identifying any thread groups that do not have the same current and home scheduling locales (Kimmel, column 10, lines 34-40). As described at one of Examiner's cited passages (Kimmel, column 10, lines 50-65), when Kimmel's medium term scheduler identifies a languishing thread group, it may do one of three things. First it may raise the thread group's priority. Secondly it may assign the thread group to the run queue of a higher node. And thirdly the medium term scheduler may set a flag associated with the thread group to allow poaching of the thread group by another JP. Thus, Kimmel's medium term scheduler has absolutely nothing to do with receiving requests from a multi-threaded application. Nor does Kimmel's medium term scheduler send the requests to the JP dispatcher, which the Examiner equates to a primary scheduler. Instead, as noted above, the medium term schedule monitors the state of various thread groups through Kimmel's thread group structure and changes priorities and scheduling locales accordingly to maintain balanced processor and memory loads.

In the Response to Arguments, the Examiner cites column 2, lines 30-45, column 3 lines 5-15, column 5, lines 43-58, column 6, lines 49-61, and again cites column 8, lines 24-32, column 9, lines 9-37, column 10, lines 50-65, and column 12, lines 56-65, without any explanation to respond to or rebut Applicants' previously presented arguments, outlined above. The Examiner merely asserts, "Kimmel teaches a secondary scheduler is executable to receive a plurality of requests from a multi-threaded application and send the requests to the primary scheduler." However, none of that passages cited by the Examiner actually describe any sort of secondary scheduler that receives requests from a multi-threaded application and sends them to a primary scheduler. Instead, the cited portion of column 2 discloses a medium term scheduler, which, as shown above, does not receive requests from a multi-threaded application and thus cannot be considered the secondary scheduler of Applicants' claim. The cited portion of column 3 describes that

each active process or thread-group has scheduling locales, e.g., home-scheduling locale and a current scheduling locale. As shown above, scheduling locales are not relevant to a secondary scheduler executable to *receive requests from a multi-threaded application* and to *send the requests to a primary scheduler.* Applicants have already shown this in previous office action responses and above, and the Examiner has failed to accurately rebut any of these arguments. The cited portions of columns 5 and 6 also fail to support the Examiner's contention. Instead, the cited portion of column 5 describes Kimmel's dispatcher and medium term scheduler while the cited portion of column 6 refers to Kimmel's dispatcher and global and local scheduling. As shown above, the dispatcher of Kimmel's system can not be considered the primary scheduler of claim 1 since Kimmel's dispatcher it does not schedule requests for network data resources sent from a secondary scheduler.

Maresco is relied upon by the Examiner only to teach "a thread safe system" and the Examiner cites column 2, lines 30-38. However, Maresco fails to overcome any of the above noted deficiencies in Kimmel regarding claim 1. Thus, the Examiner's combination of Kimmel in view of Maresco would not result in a system that includes a primary scheduler which is executable to schedule requests for networked data resources; and a secondary scheduler, wherein the secondary scheduler is executable to receive a plurality of requests from a multi-threaded application in a thread-safe manner and send the requests to the primary scheduler in a thread-safe manner. Instead, even if the references were properly combinable (which Applicants do not concede), the Examiner's proposed combination of Kimmel and Maresco would at most result in Kimmel's system for maintaining balanced processor and memory loads in a NUMA system where Kimmel's thread groups are the work crews of Maresco.

Moreover, Maresco does not teach or suggest anything about receiving or sending requests for networked data resources in a thread-safe manner. Instead, Maresco is concerned with safely *creating* threads for task execution is a computer system (Maresco, column 1, lines 42-57; column 2, lines 25-38). Maresco teaches nothing about a scheduler receiving and sending requests in a thread-safe manner. Just because Maresco

mentions thread creation for task execution does not imply that Maresco suggests receiving and sending requests by schedulers in a thread-safe manner. As discussed above, neither Kimmel nor Maresco has anything to do with scheduling requests from a multi-threaded application, let alone receiving and sending requests for scheduling in a thread-safe manner.

Applicants note that the Examiner has not responded to the arguments above relating to Maresco.

For at least the reasons above, the rejection of claim 1 is not supported by the prior art and removal thereof is respectfully requested.

Regarding claim 15, contrary to the Examiner's assertion, Kimmel in view of Maresco fails to teach or suggest receiving a plurality of management requests from a multi-threaded manager application into a secondary scheduler in a thread-safe manner. The Examiner fails to cite any portion of Kimmel or Maresco that teaches receiving a plurality of management requests from a multi-threaded manager application in the rejection of claim 15. Furthermore, as noted above regarding claim 1, neither Kimmel nor Maresco describes anything regarding receiving management requests from a multi-threaded manager application. Neither Kimmel nor Maresco are concerned with management requests from multi-threaded manager applications. Instead, Kimmel and Maresco teach respective systems for managing the execution of threads within a multi-processor computer system. For a more detailed discussion regarding Kimmel's and Maresco's failure to teach or suggest receiving requests (whether management requests or any other requests) from a multi-threaded application (whether a manager application or any other application) please see the remarks above regarding claim 1.

Additionally, Kimmel in view of Maresco fails to teach or suggest scheduling the plurality of management requests in a secondary queue in the secondary scheduler after receiving the management requests from the manager application. The Examiner cites several passages of Kimmel referring to Kimmel's medium term

scheduler. However, as discussed above, Kimmel's medium term scheduler does not receive any requests from a multi-threaded application and clearly does not receive management requests from a multi-threaded manager application. Instead, as noted above, Kimmel's medium term scheduler "monitors the progress of active processes in the system and sets a flag for those processes that are not progressing" (Kimmel, column 2, lines 40-45). Furthermore, Kimmel's medium term scheduler does not schedule management requests, but instead monitors and adjusts the priority of threads in thread groups executing on a multi-processor system.

In further regard of claim 15, Kimmel in view of Maresco also fails to teach or suggest sending the management requests from the secondary scheduler to a primary scheduler in a thread-safe manner. As noted above regarding claim 1, Kimmel fails to describe his medium term scheduler, which the Examiner equates to the secondary scheduler, as sending management requests to his dispatcher, which the Examiner equates with a primary scheduler. Instead, both the medium term scheduler and the dispatcher access Kimmel's hierarchical node structure to monitor the states of the individual threads and thread groups, adjusting the priority and scheduling policies to ensure balanced processor and memory loading. Nowhere does Kimmel describe his medium term scheduler sending management requests to the dispatcher. Maresco also fails to mention anything regarding sending management requests from a secondary scheduler to a primary scheduler. For a more detailed discussion regarding Kimmel's and Maresco's failure to teach or suggesting sending management requests from a secondary scheduler to a primary scheduler, please refer to the discussion above regarding claim 1.

Furthermore, Kimmel in view of Maresco fails to teach or suggest executing the management requests on the managed objects after scheduling the management requests in the primary queue. The Examiner cites column 6, lines 62-67 where Kimmel describes how using thread groups in developing processes give a user flexibility "to choose between creating a new thread within an existing thread group or creating a new thread group." The cited passage makes no mention of executing management

requests on managed objects. Creating thread groups and giving users the ability to choose between adding a new thread to an existing thread group or creating a new thread group has no relevance to management requests or managed objects, as they are understood in the art. As noted above, nowhere does Kimmel mention anything regarding either management requests for managed objects. Maresco is only relied upon the by the Examiner to teach a thread safe system. Maresco fails to overcome any of the above noted deficiencies of Kimmel.

Thus, the Examiner's proposed combination of Kimmel and Maresco fails to teach or suggest receiving a plurality of management requests from a multi-threaded manager application into a secondary scheduler in a thread-safe manner; scheduling the plurality of management requests in a secondary queue in the secondary scheduler after receiving the management requests from the manager application; sending the management requests from the secondary scheduler to a primary scheduler in a thread-safe manner; or executing the management requests on the managed objects after scheduling the management requests in the primary queue. Instead, as noted above regarding claim 1, the Examiner's proposed combination of Kimmel and Maresco would result in Kimmel's system for maintaining balanced processor and memory loads in a NUMA system where Kimmel's thread groups are the work crews of Maresco.

Applicants note that the Final Action the Examiner failed to provide any rebuttal to the above arguments relating to claim 15. For at least the reason given above, the rejection of claim 15 is not supported by the prior art and removal thereof is respectfully requested. Similar remarks as those above regarding claim 15 also apply to claim 27.

Regarding claim 2, Kimmel in view of Maresco fails to teach or suggest, contrary to the Examiner's assertion, wherein the primary scheduler is single-threaded. The Examiner cites column 5, lines 50-56 and column 24, lines 14-26. However, both of the Examiner's cited passages describe how Kimmel's system may be used in conjunction with thread groups including various numbers of threads, including single thread sized

thread groups. The cited passages do not mention anything regarding whether Kimmel's dispatcher, which the Examiner equates to a primary scheduler, is single-threaded. Additionally, Maresco fails to teach anything regarding wherein a primary scheduler is single-threaded and thus fails to overcome the above noted deficiency of Kimmel. Thus, the rejection of claim 2 is not supported by the prior art and removal thereof is respectfully requested.

Regarding claim 3, Kimmel in view of Maresco fails to teach or suggest wherein the secondary scheduler is multi-threaded, in contrast to the Examiner's contention. The Examiner again cites column 5, lines 50-56 and column 24, lines 14-26. However, as noted above, both of the Examiner's cited passages describe how Kimmel's system may be used in conjunction with thread groups including various numbers of threads, including single thread sized thread groups. Neither of the cited passages mention anything regarding whether Kimmel's medium term scheduler, which the Examiner equates to a secondary scheduler, is multi-threaded. Additionally, Maresco fails to teach anything regarding wherein a secondary scheduler is multi-threaded and thus fails to overcome the above noted deficiency of Kimmel. Thus, the rejection of claim 3 is not supported by the prior art and removal thereof is respectfully requested.

Regarding claim 4, Kimmel in view of Maresco fails to teach or suggest wherein secondary scheduler is executable to receive the plurality of requests from the multi-threaded application through a lock in a thread-safe manner. However, as noted above regarding claim 1, Kimmel (and Maresco) fails to teach or suggest receiving a plurality of requests from a multi-threaded application. For a detailed discussion regarding Kimmel's (and Maresco's) failure to teach receiving requests from a multi-threaded application, please see the above discussion regarding claim 1.

The Examiner cites column 11, lines 5-30 and element 144a of Fig. 8 in Kimmel. **However, the Kimmel reference does not include a FIG. 8 nor an element 144a in any Figure.** Since the same passages (column 1, lines 5-30 and element 144a of FIG. 8) where cited in Cheeseman, et al. (U.S. Patent 6,680,933) in the Final Office Action dated

December 17, 2004, applicants assume the Examiner mistakenly used the same passages citations in the current rejection of claim 4. Thus, the Examiner has failed to point out or cite any passage of Kimmel that teaches or suggests wherein secondary scheduler is executable to receive the plurality of requests from the multi-threaded application through a lock in a thread-safe manner. Nowhere does Kimmel mention anything regarding a lock through which a plurality of requests may be received in a thread-safe manner from a multi-threaded application. **Applicants note that the Examiner has failed to respond to this argument or correct the citations relied upon in the rejection of claim 4.**

The Examiner also cites column 2, lines 30-38 and column 6, lines 15-19 of Maresco and argues that Maresco teaches a thread-safe system. However, nowhere, whether at the Examiner's cited passages or elsewhere, does Maresco teach or suggest anything regarding receiving the plurality of requests from the multi-threaded application through a lock in a thread-safe manner. Thus, neither Kimmel nor Maresco, whether considered singly or in combination, teaches or suggests wherein secondary scheduler is executable to receive the plurality of requests from the multi-threaded application through a lock in a thread-safe manner.

For at least the reasons above, the rejection of claim 4 is not supported by the prior art and removal thereof is respectfully requested.

Regarding claim 5, Kimmel in view of Maresco fails to teach or suggest wherein the primary scheduler is executable to receive the plurality of requests from the secondary scheduler through a lock in a thread-safe manner. The Examiner cites column 24, lines 14-26 and column 6, lines 43-65. However, neither of the cited passage mentions anything about Kimmel's dispatcher, which the Examiner equates to a primary scheduler, being executable to receive requests from the medium term scheduler, which the Examiner equates to a secondary scheduler, through a lock in a thread-safe manner. Instead, as noted previously, The Examiner's first cited passage (column 24, lines 14-26) describes how Kimmel's operating system may function in systems having various

number of execution queue levels and that have different sized thread groups. The second passage describes the thread group structure maintaining cumulative timeslice and job processor (JP) accounting for all threads in a thread group. For a more detailed discussion regarding these passages, please see the above discussion regarding claim 1. As argued above, regarding claim 1, Kimmel in view of Maresco fails to teach or suggest a secondary scheduler sending requests to a primary scheduler. Additionally, Kimmel fails to mention anything regarding a lock through which a primary scheduler may receive requests from a secondary scheduler.

The Examiner has failed to cite any portion of Maresco regarding the rejection of claim 5. However, Applicants assume the Examiner intended to cite column 2, lines 30-38 and column 6, lines 15-19 of Maresco. However as noted above regarding claim 4, Maresco fails to teach anything regarding a lock through which requests may be received.

Thus, the Examiner's combination of Kimmel and Maresco fails to teach or suggest wherein the primary scheduler is executable to receive the plurality of requests from the secondary scheduler through a lock in a thread-safe manner. Therefore, the rejection of claim 5 is not supported by the prior art and removal thereof is respectfully requested.

Regarding claim 16, Kimmel in view of Maresco fails to teach or suggest wherein executing the management requests on the managed objects further comprises sending the management requests to a management information server coupled to the managed objects. The Examiner cites column 12, lines 7-15 and column 11, lines 23-40 of Kimmel. However, the cited passages describe how Kimmel's dispatcher, which the Examiner equates to a primary scheduler, searches for a languishing thread group to execute. No mention is made by Kimmel, either in the cited passages or elsewhere, of a management information server coupled to managed objects. Management information servers and managed objects are well understood in the art and a dispatcher searching through Kimmel's hierarchical data structure to locate a languishing through group to

execute does not have any relevance to either a management information server or to managed objects.

The Examiner does not rely upon Maresco to teach, nor does Maresco teach, anything regarding sending management requests to a management information server coupled to managed objects. Thus, the Examiner's combination of Kimmel and Maresco fails to teach or suggest wherein executing the management requests on the managed objects further comprises sending the management requests to a management information server coupled to the managed objects.

For at least the reasons presented above, the rejection of claim 16 is not supported by the prior art and removal thereof is respectfully requested. Similar remarks apply to claim 28.

Regarding claim 17, Kimmel in view of Maresco fails to teach or suggest wherein each of the management requests comprises a corresponding callback function. The Examiner cites column 3, lines 37-48 of Maresco. However, Maresco does not describe anything regarding management request comprising callback functions. The Examiner cited passage describes how Maresco's CWorker object class maintains a reference to the work crew that it belong to and how each CWorker also keeps track of the next and previous members of the worker list for the work crew. Neither Kimmel nor Maresco teaches anything that can be interpreted as management request each comprising a corresponding callback function. Thus, the rejection of claim 17 is not supported by the prior art and removal thereof is respectfully requested. Similar remarks apply to claim 29.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the rejection has been shown to be unsupported for the independent claims, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicants submit the application is in condition for allowance, and notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-48600/RCK.

Also enclosed herewith are the following items:

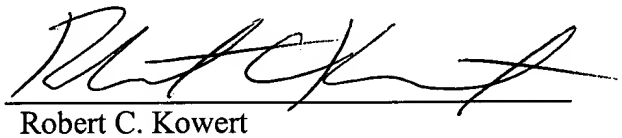
☒ Return Receipt Postcard

☐ Petition for Extension of Time

☐ Notice of Change of Address

☐ Other:

Respectfully submitted,



Robert C. Kowert
Reg. No. 39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: November 18, 2005